Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Podlesnik

Navidezno resnični sistem za interakcijo z molekulskimi strukturami na osnovi Oculus Rift očal in globinskih senzorjev

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Navidezno resnični sistem za interakcijo z molekulskimi strukturami na osnovi Oculus Rift očal in globinskih senzorjev

Implementirajte sistem za vizualizacijo molekulskih struktur po zgledu sistemov, ki jih uporabljajo kemiki. Preučite možnosti integracije Oculus Rift očal ter globinskih senzorjev Leap Motion in Kinect v ta sistem. Predstavite razvoj sistema ter primere uporabe.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Podlesnik, z vpisno številko 63080183, sem avtor diplomskega dela z naslovom:

Navidezno resnični sistem za interakcijo z molekulskimi strukturami na osnovi Oculus Rift očal in globinskih senzorjev (angl. *Virtual reality system for interaction with molecular structures based on Oculus Rift and depth sensors*)

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Petra Peera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 11. marca 2016 Podpis avtorja:

Rad bi se zahvalil mentorju izr. prof. dr. Petru Peeru za nasvete in pripombe pri izdelavi tega dela. Prav tako bi se rad zahvalil doc. dr. Črtomirju Podlipniku za pomoč pri kemijskem delu naloge. Posebna zahvala pa gre tudi mojim staršem za potrpljenje in podporo med študijem.

Kazalo

Povzetek

Naslov: Navidezno resnični sistem za interakcijo z molekulskimi strukturami na osnovi Oculus Rift očal in globinskih senzorjev

Cilj diplomskega dela je sistem za rokovanje z molekulskimi strukturami v navideznem svetu s pomočjo globinskih senzorjev. Uporabljena so bila 3D-očala Oculus Rift, senzorja Kinect in Leap Motion ter igralni pogon Unity. Razvite so bile štiri različice aplikacije. Osnovno različico, ki je uporabna brez senzorjev, za vsak senzor eno in še končno različico, ki združi vse prejšnje. Ena bolj očitnih ugotovitev je, da se več različnih senzorjev ne splača uporabiti skupaj, saj se med sabo motijo. Tako je bolje uporabiti samo en senzor in se temu bolj posvetiti.

Ključne besede: Oculus Rift, Kinect, Leap Motion, Unity, navidezna resničnost, molekula, molekulsko modeliranje, vizualizacija, interaktivnost

Abstract

Title: Virtual reality system for interaction with molecular structures based on Oculus Rift and depth sensors

The main goal of this diploma thesis is developement of a system for interacting with molecular stuctures in virtual world with the help of depth sensors. We used Oculus Rift 3D glases, Kinect and Leap Motion sensors with Unity game engine. We developed four diferent versions of the aplication. The basic version without sensors, one version for each sensor, and the final version that combines all others. One of more obvious finding is that it is better not to use more sensors at the same time, because they interfere with each other. So it is better to use only one sensor and put more work into it.

Keywords: Oculus Rift, Kinect, Leap Motion, Unity, virtual reality, molecule, molecular modeling, visualization, interactivity

Poglavje 1

Uvod

Z razvojem tehnologije se je v zadnjih letih spet prebudilo zanimanje za navidezno resničnost (angl. virtual reality). Po uspešni kampaniji Kickstarter podjetja Oculus VR za razvoj njihove ideje očal za navidezno resničnost, imenovanih The Rift ali na kratko samo Rift, pa so za to področje pokazala zanimanje tudi druga podjetja, kot so Samsung, HTC in Sony. Seveda pa pri omembi navidezne resničnosti po navadi pomislimo tudi na intuitivno interakcijo z navideznim svetom in tako pridemo do tematike senzorjev, ki zaznavajo naše gibe in jih ustrezno prenesejo v navidezni svet. In če hkrati s tem omenimo še molekulsko modeliranje, kmalu ugotovimo, da je potencial prej omenjenih tehnologij na področju kemije in sorodnih področij izjemen.

Na podlagi te ideje smo se odločili izdelati aplikacijo za interakcijo z molekulskimi strukturami. Pri tem bomo uporabili očala Oculus Rift ter senzorja Kinect in Leap Motion. Cilj tega dela je torej izdelati aplikacijo ali vsaj prototip aplikacije, ki bi bila uporabna tudi na Fakulteti za kemijo in kemijsko tehnologijo za izobraževalne ali raziskovalne namene. S tem bomo ugotovili, koliko in kakšna znanja potrebujemo, koliko časa potrebujemo in ali se takšna aplikacija z dano tehnologijo da narediti in ali se splača.

V naslednjih poglavjih bosta podrobno predstavljena tematika in potek dela. V drugem poglavju je predstavljeno teoretično ozadje navidezne resničnosti in nekaj nekaj kemijskega ozadja o molekulskih strukturah. V tre-

tjem poglavju predstavimo sorodna dela, med katerimi izstopa Molecular Rift. V četrtem poglavju so podrobno predstavljene uporabljene tehnologije in orodja, torej očala Oculus Rift, senzorja Kinect in Leap Motion ter igralni pogon Unity 3D. V petem poglavju je predstavljen razvoj aplikacije, nato pa sledijo še primeri uporabe v šestem poglavju ter zaključek v sedmem.

Poglavje 2

Teoretično ozadje

2.1 Navidezna resničnost

Navidezna resničnost (angl. virtual reality) je oblika računalniške simulacije, kjer poskušamo umetno simulirati neko resnično ali izmišljeno okolje. Torej je virtualna resničnost simulacija vizualnih, zvočnih in drugih čutnih zaznav z namenom prepričati naša čutila, da smo nekje drugje [1].

2.1.1 Zgodovina

Prve omembe navidezne resničnosti so se pojavile v znanstvenofantastičnih literarnih delih. Navidezna resničnost, kot jo poznamo danes, pa se je začela razvijati konec petdesetih in v začetku šestdesetih let prejšnjega stoletja. Leta 1962 je Morton Heilig zgradil prototip, v katerem si je lahko uporabnik ogledal pet kratkih filmov s spremljajočim zvokom, premikanjem stola in celo vonjem (slika 2.1).

Leta 1966 je Thomas A. Furness III predstavil navidezno resničnost ameriški vojski v obliki simulatorja pilotiranja letala.

Tehnologija se je počasi razvijala skozi leta in doživela vzpon v devetdesetih letih z veliko inovacijami ter razvojem iger in igranih pripomočkov, kot so veliki naglavni zasloni, rokavice, razne igralne palice in drugo. Vendar so se kljub novosti tehnologije ter zanimivosti le-te, izdelki povezani z navidezno

resničnostjo kmalu poslovili, saj so bili velikokrat nezanesljivi, preokorni in predragi. Področje je bilo bolj ali manj pozabljeno naslednjih 20 let, dokler se ni pojavil Oculus Rift. Ta je sprožil nov razcvet na področju navidezne resničnosti, saj so izdelku kmalu sledila podjetja Google, Samsung, HTC in Sony s svojimi rešitvami [2].

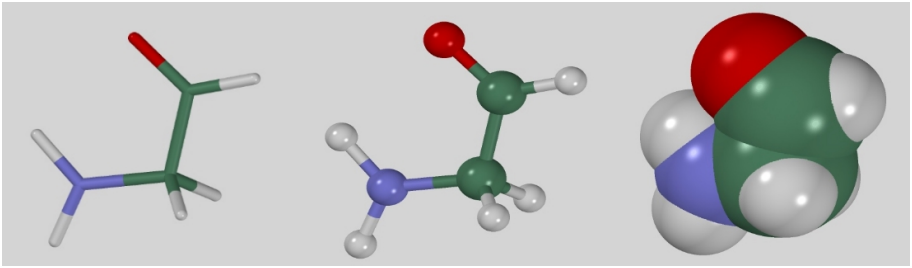
Slika 2.1: Navidezno resnični sistem Sensorama iz leta 1962

2.2 Molekulske strukture, modeliranje in vizualizacija

Molekulski model je predstavitev molekule. Koncept molekul kot 3D-objektov se je pojavil nekje v 19. stoletju. Prvo prestavitev krogličnega (angl. ball-and-stick) modela pripisujejo nemškemu kemiku Augustu Wilhelmu von Ho-

fmannu, ki je leta 1865 uporabil take modele med predavanjem na kraljevski inštituciji Velike Britanije (angl. Royal Institution of Great Britain) [3].

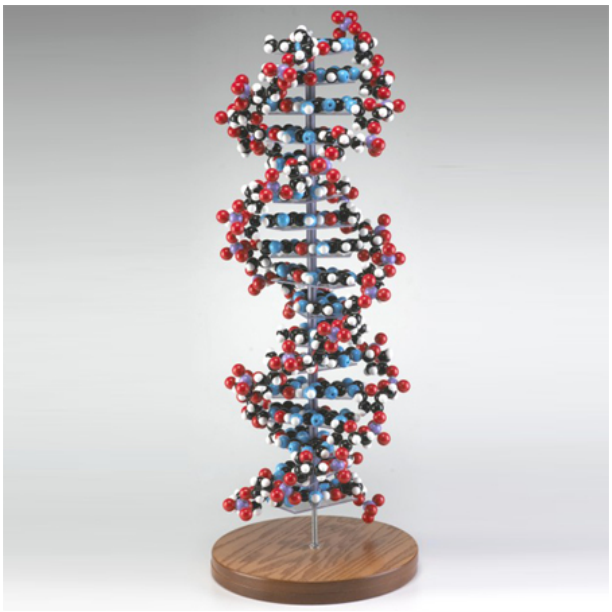
Dandanes poznamo več načinov prikaza z modeli, za nas so pomembni palični ali skeletni (angl. stick) model, krogličnі (angl. ball-and-stick) model in kalotni (angl. space filling) model [4, 5], ki so predstvljeni na sliki 2.2 z leve proti desni.



Slika 2.2: Vrste molekulskih modelov

Molekulski modeli so skozi zgodovino igrali pomembno vlogo pri razumevanju kemije in pri postavljanju ter testiranju hipotez. Sami molekulski modeli pa so oblikovani na podlagi znanstvenih raziskav, s katerimi je omogočeno neposredno zaznavanje submikroskopskega sveta. Do molekulskega modela pridemo s pomočjo eksperimentalnih ali računalniških metod. Med eksperimentalne sodita kristalografija (angl. X-ray crystallography) in nuklearna magnetna resonanca, med računalniške pa računalniško molekulska mehanika [6] in kvantno kemijske metode.

Predstavitev kompleksnih molekulskih struktur s pomočjo modela je zelo pomembna pri proučevanju znanstvenoraziskovalnih problemov. Znanstvenikom namreč omogoča proučevanje pojavov na molekularni in atomski ravni, zaradi česar sta molekulsko modeliranje in vizualizacija nepogrešljivi orodji raziskovalcev na kemijskem in sorodnih področjih. Zgodovinsko gledano je eden izmed najbolj znanih molekulskih modelov model dvojne vijačnice strukture DNA (slika 2.3).



Slika 2.3: Molekulski model DNA

Molekulski modeli pa se seveda že dolgo uporabljajo tudi v izobraževalne namene. Za pomoč pri poučevanju in predstavitvi svojih idej je svoje modele uporabljal že John Dalton leta 1811. Raziskave pa so potrdile, da uporaba molekulskih modelov kot pripomočkov pri učenju izboljša razumevanje snovi [7]. Tako kot fizični modeli so enako uporabni tudi računalniški, mogoče še bolj, saj nudijo veliko dodatnih možnosti.

2.2.1 Vizualizacija v molekulskem modeliranju

S časom in razvojem tehnologij se je izboljšala predstavitev 3D-modelov molekul. Še posebej je k temu pripomogla rentgenska tehnologija. Fizične predstavitve molekul so bile v veliko pomoč pri predstavitvi in analizi molekul, s čimer so prispevale k mnogim pomembnim odkritjem.

Z razvojem računalniške tehnologije pa se je vizualizacija s fizičnih modelov počasi prenesla na simulirane računalniške modele. S časom in tehnološkim razvojem je postalo mogoče prikazati te modele s tehniko stereoskopije. Večina modernih orodij za vizualizacijo in molekulsko modeliranje,

kot so YASARA, PyMol, Chimera in drugi, uporablja tudi to metodo.

Stereoskopija je tehnika obdelave, ki uporabniku omogoča videti tretjo dimenzijo [8]. To je doseženo tako, da sliko ali posnetek posnamemo pod rahlo različnim kotom, to potem združimo in pogledamo s posebnimi očali.

Lahko jo delimo na dva tipa:

- pasivna stereoskopija,
- aktivna stereoskopija.

Osnovna ideja pasivne stereoskopije je, da sliko izrišemo dvakrat v kromatično nasprotujočih si barvah, na primer v rdeči in modri, pri čemer mora biti ena slika malo zamaknjena. Nato si sliko ogledamo s posebnimi očali (slika 2.4).

Aktivna stereoskopija pa uporabi drugačen pristop. Tukaj potrebujemo posebna 3D-očala z lečami iz tekočih kristalov, ki potem glede na signal, ki ga dobijo od naprave, na katero so priklopljene, preprečijo prikaz slike enemu očesu. Z izmeničnim prikazom slike na levem in desnem očesu se ustvari iluzija tretje dimenzije (slika 2.5).

Slika 2.4: Pasivna stereoskopska 3D-očala

Slika 2.5: Brezžična aktivna stereoskopska očala z vgrajeno baterijo

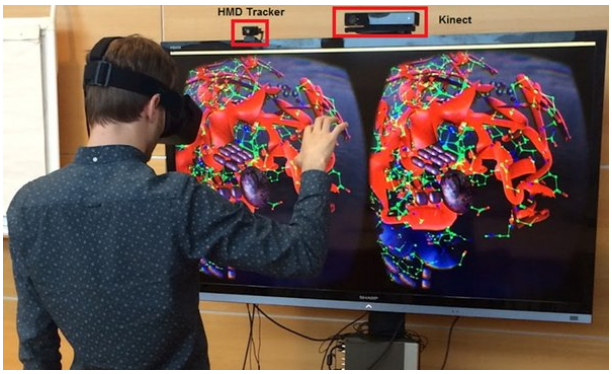
Poglavje 3

Sorodna dela

3.1 Molecular Rift

Dandanašnji hiter razvoj v računalništvu in na področjih vizualizacije podatkov ter interakcije z napravami je omogočil prenos marsikatere ideje v realnost. Ena od teh je prikaz raznih objektov v navidezno resničnih svetovih ter interakcija z njimi. Vizualizacija molekulskih struktur je pomembna za moderno farmacevtsko industrijo. Dobra vizualizacijska metoda pripomore pri analizi, razumevanju in sprejemanju odločitev pri razvoju zdravil. Tako je nastal Molecular Rift magistrsko delo v sodelovanju s farmacevtskim podjetjem AstraZeneca. Cilj je bil raziskati nove vizualizacijske in interakcijske metode ter ugotoviti njihovo uporabnost [9].

Aplikacija je narejena v igralnem pogonu Unity 5 Personal Edition, za prikaz navideznega okolja so uporabljena očala Oculus Rift Development Kit 2, za zaznavanje gest pa senzor Kinect druge generacije. Koda je večinoma napisana v programskem jeziku C# in nekaj malega v C++. Vgradili so tudi povezavo z orodjem Open Babel za pridobivanje kemijskih informacij. Med razvojem je bilo razvito več iteracij aplikacije, ki so se spreminjale glede na povratne informacije, ki so bile dobljene s strani uporabnikov (slika 3.1).



Slika 3.1: Uporaba aplikacije Molecular Rift

3.2 HeroVR

Da bi dosegli čim večji občutek realizma navideznega sveta, je potrebna povratna informacija. Z vprašanjem se bolj ali manj ukvarja že nekaj izdelkov na tržišču, HeroVR pa se ukvarja s tem, kako zagotoviti sledenje gibanja brez dodatnih specializiranih senzorjev ali markerjev. HeroVR, razvit v sodelovanju dveh podjetij, Agency for Virtual Reality (A4VR) in The Captury, omogoča s pomočjo osmih do šestnajstih kamer v prostoru in očal Oculus Rift gibanje v izbranem virtualnem okolju (slika 3.2) [10].

Slika 3.2: Predstavitev delovanja sistema HeroVR

Poglavje 4

Tehnologije in orodja

4.1 Oculus Rift

Oculus Rift so naglavna očala za navidezno resničnost, razvita v podjetju Oculus VR. Projekt se je sprva predstavil na Kickstarterju in se po uspešno doseženih ciljih kampanije samo še razvijal. Kasneje je bilo podjetje Oculus VR prodano Facebooku. Komercialna različica očal Rift je na voljo preko prednaročila, naročniki pa jih bodo dobili od 28. marca dalje [11]. Med razvojem je podjetje izdalo dva razvojna paketa (angl. development kit), ki sta bila namenjena predvsem razvijalcem programske opreme za prihajajočo komercialno verzijo in kot prototip za testiranje. Za prikaz učinka 3D uporabljajo očala Rift leče in dva OLED-zaslona ločljivosti 960 x 1080. Vsebuje tudi giroskop, merilec pospeškov in magnetometer za zaznavanje orientacije ter obračanja. Zraven je dodana tudi manjša zunanja kamera za zaznavanje položaja očal v prostoru, ki pa sicer za uporabo ni potrebna, je pa priporočljiva. Očala Rift za dobro delovanje potrebujejo sistem Windows, zmogljivo centralno procesno enoto ter samostojno in zmogljivo grafično kartico. Očala ne delujejo na prenosnikih, razen nekaj izjem [12, 13]. Različica, ki smo jo uporabili za potrebe diplomskega dela, je Development kit 2 (slika 4.1) s knjižnico SDK za Windows verzije 0.8.

Slika 4.1: Očala Oculus Rift

4.2 Globinski senzorji

4.2.1 Kinect

Kinect je krmilnik gibanja podjetja Microsoft za platforme Xbox 360, Xbox One in Windows. Na računalnik ga priključimo preko vmesnika USB. Videti je kot malo večja podolgovata spletna kamera črne barve, ki jo po navadi postavimo nad ekran ali zraven njega. Kinect omogoča interakcijo z računalnikom ali s konzolo brez uporabe tipkovnice in miške ali igralnega ploščka preko naravnega uporabniškega vmesnika (angl. natural user interface). To doseže z zaznavanjem kretenj ali prepoznavanjem glasovnih ukazov.

Kinect prve generacije je bil sprva predstavljen konec leta 2010 kot dodatek Xboxu 360 z namenom povečanja števila uporabnikov oziroma razširitev občinstva iz tipične skupine ljudi, ki igrajo igre. Različica za Windows je izšla v začetku leta 2012. Nadgrajena različica druge generacije je izšla skupaj z igralno konzolo Xbox One konec leta 2013. Za Windows prilagojena različica je izšla sredi leta 2014 skupaj s knjižnico Kinect za Windows SDK 2.0.

Za zaznavanje ukazov ima Kinect vgrajeno RGB-kamero, infrardeči oddajnik in infrardečo globinsko kamero ter polje štirih mikrofonov. S pomočjo teh senzorjev se prejeti podatki s pomočjo določenih algoritmov pretvo-

rijo v signale, ki jih potem lahko programsko pretvorimo v ukaze, ki jih računalnik prepozna. Čeprav je bil Kinect sprva tržen kot igralni pripomoček, se je zaradi ugodne cene in dobavljivosti kmalu razširil tudi na druga področja [14, 15].

Za potrebe diplomskega dela smo uporabili senzor Kinect prve generacije (slika 4.2) s knjižnico SDK za Kinect za Windows verzije 1.8.

Slika 4.2: Senzor Kinect

4.2.2 Leap Motion

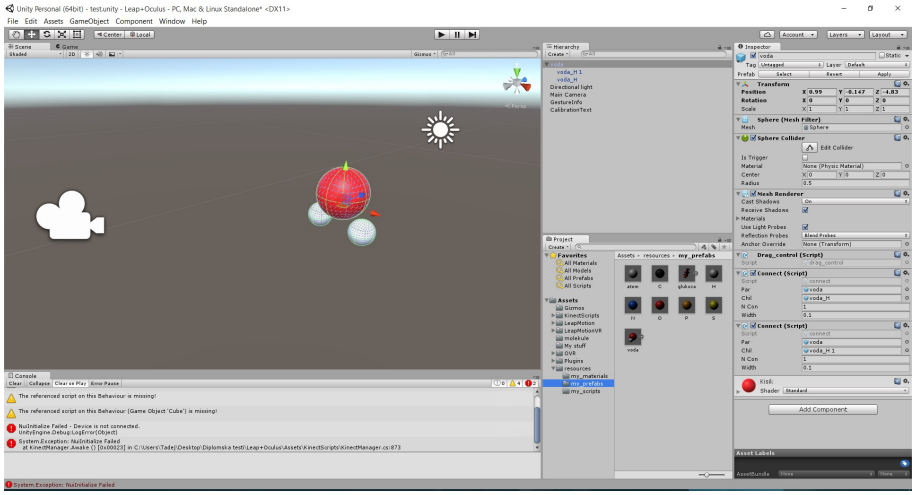
Leap Motion je računalniška strojna oprema, ki podpira zaznavanje gibanja dlani in prstov za brezdotični vnos podatkov v računalnik, namesto miške in tipkovnice. Naprava je majhna, moderne oblike in z računalnikom povezana preko USB-vmesnika. Za zajem gibanja vsebuje tri infrardeče LED-luči in dve infrardeči kameri. Omogoča zajem gibanja v polkrogelnem prostoru nad senzorjem do razdalje enega metra. Zaradi manjšega območja zaznavanja in večje ločljivosti naprave se razlikuje od Kinecta, ki je bolj namenjen zaznavi celega telesa, je pa zato bolj natančen pri zaznavi gibov prstov [16]. Za potrebe diplomskega dela smo uporabili senzor Leap Motion (slika 4.3) s knjižnico SDK za Windows verzije 2.3.

Slika 4.3: Senzor Leap Motion

4.3 Unity 3D

Razvoj in testiranje aplikacije za potrebe diplomskega dela je potekalo v okolju Windows 8.1 znotraj igralnega pogona Unity 3D (slika 4.4) [17]. Unity 3D je igralni pogon, razvit v podjetju Unity Tehnologies, in v osnovi namenjen razvoju iger na več platformah, kot so Windows, OS X, konzole in mobilne naprave. Njegovi dobri lastnosti sta predvsem prenosnost med napravami in prilagodljivostjo. Obstajat več različic razvojnega okolja, ki jih lahko naložimo z uradne strani, za posameznika, ki ni podjetje ali izobraževalna ustanova, pa sta pomembni predvsem dve različici: brezplačna Personal in plačljiva Professional. Razlikujejo se samo v tem, da ima plačljiva različica več funkcionalnosti in malce boljšo podporo. Del igralnega pogona Unity, ki je zadolžen za platformo Windows temelji na knjižnici OpenGL. Za uporabo Unityja namesto osnovnega OpenGLa smo se odločili, ker ima Unity že vgrajeno fiziko in osnovne modele, kot so kocke, krogle, valji in ostalo. Poleg tega pa omogoča enostaven uvoz dodatnih modelov, skript in osnovnih primerov z njihove uradne strani. Lepo je urejen tudi njihov spletni portal Asset Store [18], kam lahko naložimo svoje 3D-modele, skripte, zvočne efekte ali kar cele projekte. Tam lahko dostopamo tudi do izdelkov drugih ljudi, ki so

lahko brezplačni ali plačljivi. Za pisanje skript lahko uporabimo jezike C#, JavaScript ali Boo. Skripte lahko pišemo v razvojnih okoljih MonoDevelop, ki je priložen starejšim različicam Unity 3D, zadnje verzije pa imajo priložen zastonjsko verzijo Visual Studio Community. Dodaten razlog za uporabo Unityja je tudi ta, da Unity Technologies na svojih spletnih straneh ponuja obsežno dokumentacijo in razne programske vodiče. Če pa tu ne najdete odgovora pa se lahko vedno obrneš na uradne forume, kjer lahko vprašate obsežno in prijazno skupnost razvijalcev, ki uporabljajo Unity. Poleg tega pa tako Leap Motion kot Oculus Rift ponujata podporo za Unity, podporo za Kinect pa lahko pridobimo z Asset Stora. Za potrebe diplomskega dela smo uporabili Unity 3D za Windows verzije 5.3.



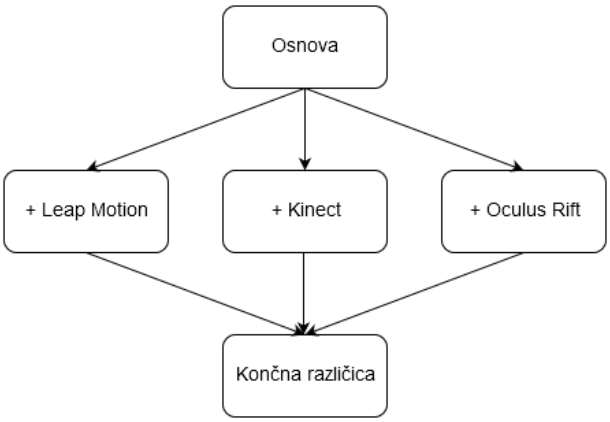
Slika 4.4: Igralni pogon Unity

Poglavje 5

Razvoj aplikacije

5.1 Osnovna vizualizacija

Ideja diplomskega dela je prikaz in interakcija z molekulskimi strukturami v 3D-prostoru, zato smo morali najprej implementirati prikaz molekulskih struktur, nato smo osnovo nadgrajevali, kot prikazuje slika 5.1.



Slika 5.1: Diagram razvoja

Najprej smo poskušali sestavi molekule iz primitivnih gradnikov, kot so krogle in valji, ter jih uporabiti za predstavitev molekul in vezi med njimi. Tukaj smo naleteli na prvo težavo, in sicer postavljati vezi med molekulami,

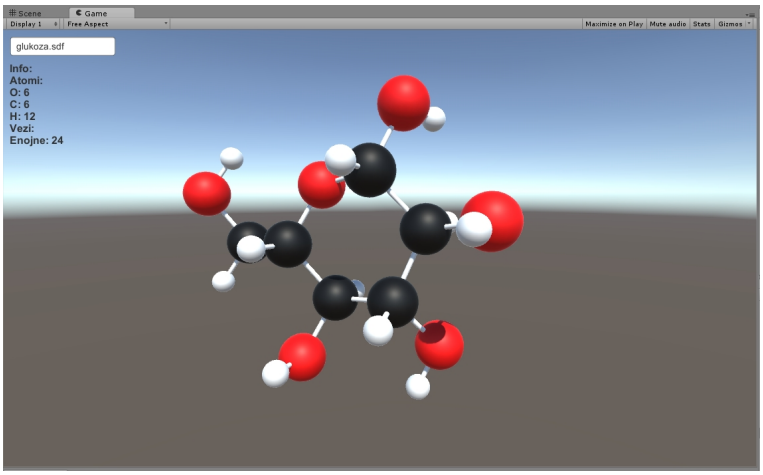
kar je zamudno in nenatančno, zato smo napisali skripto *connect.cs*, ki skrbi za to, da ustvari valj med središčema dveh objektov. To naredimo tako, da najprej določimo dva objekta, ki ju želimo povezati, nato iz teh objektov preberemo njuna središča, iz katerih potem izračunamo vektorsko razliko med njima. V Unityju so namreč točke podane kot vektorji v 3D-prostoru. S pomočjo tega novega vektorja potem izračunamo središče valja, ki ga bomo postavili med prej podana dva objekta. Položaj objektov v prostoru je namreč določen z njihovimi središči. Nato potrebujemo le še pravilno usmerjenost novega valja in skripta je končana. Usmerjenost določimo s pomočjo tistega vektorja razlike, ki smo ga izračunali na začetku, in sicer nadomestimo vektor objekta, ki kaže navzgor, glede na lokalni koordinatni sistem valja. Ta pristop deluje zato, ker je v središču vsakega objekta določen lokalni koordinatni sistem tega objekta in mi nato zasukamo celoten koordinatni sistem, s tem pa tudi objekt. Valju smo seveda morali določiti tudi pravo dolžino, dodali pa smo tudi možnost spreminjanja debeline vezi. Vse skripte smo napisali v programskem jeziku C# v prosto dostopnem razvojnem okolju Visual Studio Community.

Ko smo zagotovili enostavno ustvarjanje vezi med objekti, smo ugotovili, da je postavljanje vsakega atoma v sceno preveč zamudno, še posebej pri večjih molekulah. Zato smo tudi tu napisali skripto *ReadMol.cs*, ki sama sestavi molekulo iz danih podatkov. Najprej smo morali ustvariti predloge za posamezne atome. Te predloge (angl. prefab) so objekti, ki smo jih priredili za naše potrebe in jih potem shranili za nadaljnjo uporabo. Na primer, za določen atom najprej uporabimo gradnik kroglo, ki ji nato določimo velikost, barvo in ime ter shranimo kot predlogo tega atoma. To predlogo lahko kasneje uporabimo v skripti, kar je bolj enostavno, kot pa za vsako kroglo določati vse lastnosti programsko. Posamezne atome smo pobarvali po barvnih načelih CPK (tabela 5.1), ki so jih določili kemijki Robert Corey, Linus Pauling in Walter Koltun. Podatke o določeni molekuli smo pridobili iz datotek *.sdf* ali *.pdb*, ki smo jo dobili s spletnega portala Pubchem [19]. V skripti *ReadMol.cs* smo potem prebrali podatke in izrisali ustrezno molekulo (slika 5.2).

Oznaka	Atom	Barva
H	vodik	bela
C	ogljik	črna
O	kisik	rdeča
N	dušik	modra
S	žveplo	rumena
P	fosfor	oranžna

Tabela 5.1: Barve atomov

Nato smo morali realizirati še premikanje kamere po prostoru in interakcijo z atomi molekule. Za premikanje kamere smo uporabili osnovno skripto iz starejše verzije Unityja, imenovano *MouseLook.cs*, ki omogoča gledanje po prostoru glede na premikanje miške. To skripto smo potem preuredili za naše potrebe. Dodali smo premikanje kamere s tipkami WASD in omejili premikanje kamere samo, če pritiskamo desni miškin gumb. Za interakcijo smo dodali skripto *drag_control.cs*, ki omogoča premikanje atoma, nad katerim je miškin kazalec, po prostoru, če pritiskamo levi miškin gumb.



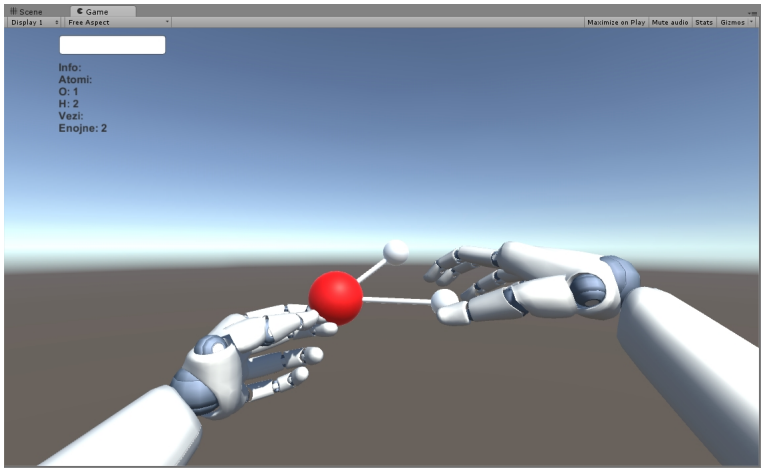
Slika 5.2: Testiranje aplikacije: molekula glukoze

5.2 Nadgradnja s senzorjem Leap Motion

Naslednji korak razvoja aplikacije je vseboval nadgradnjo tega, kar smo imeli narejeno do sedaj, s tem da smo dodali funkcionalnost premikanja izrisanih atomov in molekul z gibi, ki jih zaznamo s senzorjem Leap Motion.

Najprej smo z uradne strani Leap Motion [20] pridobili dodatke, ki so potrebni za delovanje senzorja v Unityju *(Unity Assets for Leap Motion)*. S pomočjo navodil z uradne strani in iz priloženih primerov smo potem implementirali zaznavo rok in prstov ter njihov prikaz v prostoru. To smo naredili tako, da smo vzeli že obstoječo predlogo iz paketa, ki smo ga uvozili za delovanje Leap Motiona, in to predlogo nato priredili za naše potrebe. Tako smo dobili aplikacijo, v kateri iz datoteke preberemo molekulo, jo izrišemo v prostor in nato lahko z njo rokujemo ali z miško ali pa z gestami prijemanja in vlečenja, ki jih zaznamo preko senzorja. V tej različici je mišljeno, da senzor leži na mizi pred tipkovnico (slika 5.3).

Ugotovili smo, da ta različica dobro deluje in nima izrazitih težav, dokler pazimo, da držimo roke v območju nad senzorjem. Zaznavanje v robnih območjih namreč ni najboljše.



Slika 5.3: Testiranje aplikacije: uporaba senzorja Leap Motion z molekulo vode

5.3 Nadgradnja s senzorjem Kinectom

Tudi tukaj smo nadaljevali iz osnovne različice, ki smo jo razvili in nato nadgradili z možnostjo interakcije z zaznavanjem gest, zaznanih s Kinectom.

Razvoj je potekal tako, da smo najprej s spletne strani *Asset Store* pridobili paket, ki omogoča uporabo Kinecta v Unityju s pomočjo Microsoft SDK knjižnice. Nato smo se, podobno kot pri Leap Motionu, s pomočjo priloženih navodil in primerov naučili, kako uporabiti dano ogrodje za naš primer. Ker Kinect prve generacije ni tako dober pri zaznavanju prstov in ker smo se odločili, da bomo za to uporabili že senzor Leap Motion, se v tej verziji aplikacije nismo ukvarjali z interakcijo z atomi in molekulami. Zato smo tukaj realizirali interakcijo z okoljem, torej premikanje v prostoru, ter rotacijo kamere. Da bi to dosegli, smo morali napisati oziroma dodati v že obstoječe ogrodje dve skripti, ki zaznavata geste in nato povesta, kaj se ob določenem gibu zgodi. Ti skripti sta *MyGestureListener.cs* in *GestureUse.cs*. Tu smo sprogramirali, da se ob dvigu roke kamera prestavi navzgor, ob spustu roke navzdol, ob zamahu ustrezne roke pa se kamera vrti levo ali desno. Naprej se premaknemo, če povlečemo roko k sebi, nazaj pa, če potisnemo roko stran od sebe.

Pri testiranju smo ugotovili, da aplikacija lepo deluje, vendar ima Kinect nekaj svojih težav. Za dobro delovanje mora uporabnik stati v oddaljenosti od enega do dveh metra od senzorja. Senzor mora biti postavljen dovolj visoko, drugače ne zajame celega telesa, in včasih je treba kar nekaj časa mahati, da te senzor sploh zazna. Najbolj moteče pa je verjetno to, da včasih med zaznavanjem geste le-te zazna napačno.

5.4 Končna različica

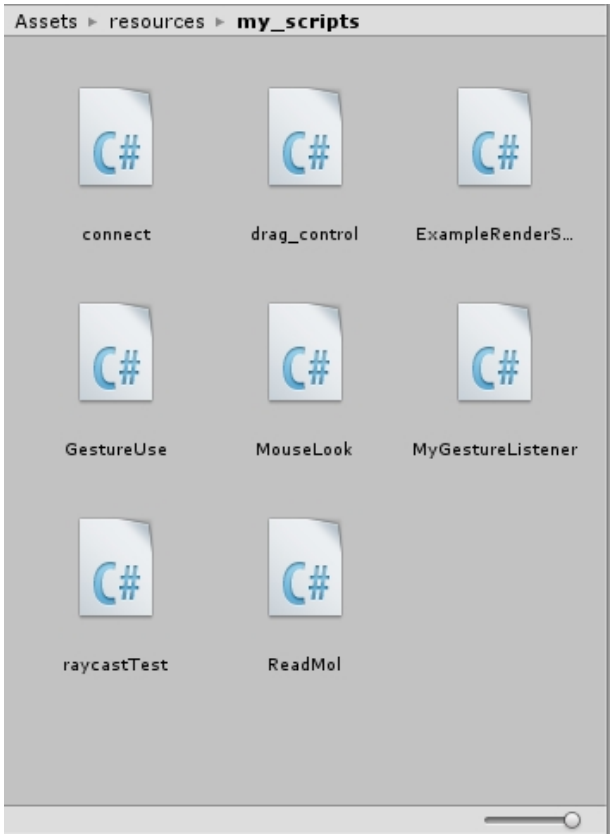
V končni različici smo združili prejšnje verzije in k temu dodali še podporo za navidezno resničnost, torej očala Rift. S tem smo dosegli cilj diplomskega dela, ki je interakcija z molekulskimi strukturami v navideznem svetu s podporo globinskih senzorjev.

Uporaba očal Rift v Unityu je zelo preprosta, saj je v različici 5.3 podpora za 3D-očala že vgrajena v program, in sicer v nastavitvah samo vklopimo možnost VR support in deluje. Tukaj pa se pojavijo tudi težave, saj očala delujejo tako, da ustvarjajo 3D-učinek s pomočjo dveh zaslonov, v sceni pa imamo samo eno kamero. Prva stvar, kjer nastopijo težave, je premikanje kamere, sej se sedaj kamera vrti glede na to, kakšne signale dobi iz premikanja 3D-očal. Zato smo jo morali, če smo hoteli kamero premikati še kako drugače, prilepiti na prazen objekt in nato ta objekt premikati s skripto. Druga težava, ki nastopi, je, da v očalih ne vidimo več grafično uporabniškega vmesnika, ki smo ga naredili, in je v prejšnjih različicah deloval normalno. To popravimo tako, da ravnino (angl. canvas), na katerega izrisujemo uporabniški vmesnik, spremenimo tako, da mu spremenimo Render mode v World space. Najbolj nadležna težava je ta, da metanje žarkov (angl. ray casting) ne deluje oziroma deluje z napako, razen če gre žarek iz središča kamera v središče zaslona. Ta težava za enkrat še ni rešena, upamo pa, da jo bodo razrešili razvijalci Unityja v kasnejših različicah.

Sledi dodatek senzorjev Leap Motion in Kinect. Ideja je bila, da naredimo prototip aplikacije, pri kateri svet gledamo skozi očala Rift, za interakcijo z objekti in premikanje pa je poskrbljeno s pomočjo omenjenih senzorjev. Delujočo aplikacijo nam je sicer uspelo izdelati, njena dejanska uporabnost pa je vprašljiva. Namreč, naleteli smo na več težav. Največja težava je verjetno ta, da več senzorjev poslabša delovanje aplikacije, namesto da jo izboljša. Kot smo povedali pri opisu senzorjev, oba uporabljata infrardečo svetlobo za globinsko zaznavanje, kar povzroči, da morata senzorja porabljati dodatne vire za preračunavanje vhodnih podatkov zaradi dodatnih virov infrardeče svetlobe ali pa javita napake. Poleg tega se velikokrat zgodi, da en senzor zazna gibe, ki so namenjeni drugem senzorju, in izvrši ukaz, ki ga sicer nismo nameravali storiti. Če izklopimo enega od senzorjev, pa aplikacija postane bolj uporabna, vendar moramo v tem primeru del ukazov nadomestiti s tipkovnico.

Ker uporaba očal Rift porabi veliko procesorskih virov, lahko izboljšamo zmogljivost aplikacije, če spremenimo kakovost izrisovanja. To storimo z uporabo kratke skripte, imenovane *ExampleRenderScale.cs*. Vse uporabljene skripte se nahajajo v Assets\resources\my_scripts (slika 5.4).

Po končanem razvoju lahko aplikacijo še zgradimo za želeno platformo, v našem primeru Windows, nato lahko aplikacijo zaganjamo kot izvršljivo datoteko.



Slika 5.4: Vse skripte kot videne v igralnem pogonu Unity

Poglavje 6

Primeri uporabe

Da ilustriramo delovanje aplikacije, lahko predstavimo nekaj primerov uporabe. V nadaljevanju bomo predstavili nekaj problemov ali področij, kjer bi lahko uporabljali našo aplikacijo, ali pa rešitve, ki smo jih uporabili v aplikaciji. Na koncu bomo podali še primer, kjer bomo razložili, kako lahko uporabimo rezultate za nadaljnji razvoj.

Primer 1: 3D-predstavitev molekulskih struktur

Veliko programov za vizualizacijo in modeliranje molekulskih struktur, kot so YASARA, Chimera, PyMol in drugi, se ukvarja s problemom realnega prikaza molekulskih struktur v 3D-obliki. Velike molekule z veliko atomi si je namreč težko predstavljati na 2D-ekranu. Dobra predstava sestave in delovanja molekulskih struktur je namreč zelo pomembna za raziskovalce v današnji farmacevtski industriji, zato se ti programi poslužujejo raznih metod, kot je uporaba pasivnih ali aktivnih stereoskopskih očal, ali pa kakšnih drugih dražjih in bolj eksotičnih metod. Ta problem naša aplikacija reši že preprosto z uporabo očal Rift za navidezno resničnost.

Primer 2: Intuitivna interakcija z molekulami brez uporabe tipkovnice ali miške

Naravna interakcija z objektiv v 3D-svetu vsekakor ni nova ideja in je bila

do sedaj izvedena že velikokrat pri različnih aplikacijah z uporabo raznih senzorjev. Naša aplikacija zaenkrat omogoča samo vizualizacijo in preprosto interakcijo z molekulskimi strukturami. Ideja je, da bi našo nadgrajeno aplikacijo uporabili tudi pri molekulskem modeliranju. Če imamo dva atoma in ju potem zgrabimo vsakega s svojo roko ter ju premikamo skupaj ali narazen, pri tem lahko z ustreznimi kemijskimi algoritmi ugotavljamo pri kakšni razdalji se pojavi vez med njima ali pri kakšni razdalji se vez pretrga, kakšna je energija pri tem in kakšen je upor atomov. Zelo dobra nadgradnja takih primerov bi bila z uporaba rokavic Gloveone [21], ki dajejo haptično povratno informacijo.

Primer 3: Uporaba za izobraževalne ali raziskovalne namene

Malo bolj dodelano različico naše aplikacije bi lahko uporabljali za izobraževalne namene. Prikaz molekul v navideznem svetu, s katerim lahko rokujemo, je neprimerno bolj zanimiv način učenja kot branje podatkov s table ali učbenika in kar je bolj zanimivo, si učenci lažje zapomnijo. Glede uporabe za raziskovalne namene pa obstaja možnost, če imamo dovolj kemijskega znanja, da nadgradimo aplikacijo tako, da upošteva vse kemijske in fizikalne zakonitosti in jo lahko potem uporabljamo kot pripomoček za poizkuse, ki jih ponavadi ne bi delali v laboratoriju; na primer, kaj se zgodi z neko molekulo, če razmaknemo dva atoma in pretrgamo neko vez.

Poglavje 7

Zaključek

V tem diplomskem delu smo naredili več verzij aplikacije za interakcijo z molekulskimi strukturami v navideznem svetu s pomočjo globinskih senzorjev. Iz tega smo sestavili končni prototip aplikacije. S primerjavo smo ugotovili, kaj deluje in kaj ne, kaj se splača uporabiti, če bi se odločili zgraditi komercialno uporabno aplikacijo. Med razvojem in z uporabo končane aplikacije smo ugotovili, s kakšnimi težavami se srečamo pri uporabi določenih tehnologij, ki smo jih uporabili pri izdelavi naše aplikacije.

Težava, na katero smo naleteli pri uporabi očal Rift, je, da jih ne morejo uporabljati vsi, namreč, nekaterim ljudem postane ob uporabi slabo. Nadaljnja težava so sistemske zahteve naprave, na kateri poganjamo Oculus Rift. Upoštevati pa moramo še probleme, na katere smo naleteli pri implementaciji v razvojnem okolju Unity, pri čem pa moramo počakati, da jih odpravijo razvijalci, predvsem je nadležna težava sledenje žarkom.

Ugotovili smo tudi, da če se odločimo za uporabo globinskih senzorjev, je bolje uporabiti enega kot pa dva različna, saj pride do motenj med njima. Ugotovili smo tudi, da je bolje, če sami sprogramiramo, kakšne geste bomo zaznavali z globinskimi senzorji, kot pa da se zanašamo na že v naprej podane predloge s strani razvijalcev. Čeprav bomo pri tem imeli več dela, bomo imeli tudi več svobode in geste bomo lahko bolj natančno določili, kar bo povzročilo, da bo končna uporabnost aplikacije boljša.

Obstaja pa še veliko možnosti za popravke in izboljšave aplikacije, kot so:

- Poleg same interakcije z molekulskimi strukturami lahko dodamo še molekulsko modeliranje.
- Z ustreznim kemijskim znanjem lahko dodamo razne nove funkcionalnosti.
- Implementiramo vgradnjo pretvornika med raznimi formati, v katerih lahko dobimo molekulske strukture.
- Implementiramo direkten dostop do interneta, namesto da moramo za vsako molekulsko strukturo prenesti datoteko z interneta v določeno mapo.

Slike

Literatura

[1] (2016) Virtual reality – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Virtual_reality

[2] (2016) The complete guide to virtual reality in 2016. Dostopno na: http://www.polygon.com/2016/1/15/10772026/virtual-reality-guide-oculus-google-cardboard-gear-vr

[3] (2016) Molecular model – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Molecular_model

[4] (2016) Molecular geometry – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Molecular_geometry

[5] (2016) i-Učbeniki, Kemija 8. Dostopno na: https://eucbeniki.sio.si/kemija8/932/index5.html

[6] Alan Hinchliffe, *Modelling molecular structures, 2nd Edition*, Chichester, England, John Wiley & Sons, 2000

[7] (2016) Uporaba molekulskih modelov. Dostopno na: http://keminfo.pef.uni-lj.si/uporaba_modelov/Uvod.htm

[8] (2016) Stereoscopy – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Stereoscopy

[9] Molecular Rift: Virtual Reality for Drug Designers, J. Chem. Inf. Model., 2015, 55 (11), pp 2475–2484. Dostopno na: http://pubs.acs.org/doi/abs/10.1021/acs.jcim.5b00544

[10] (2016) Virtual Reality Without Awkward Body Sensors. Dostopno na: http://motherboard.vice.com/read/virtual-reality-without-awkward-body-sensors

[11] (2016) Oculus Blog. Dostopno na: https://www.oculus.com/en-us/blog/oculus-rift-pre-orders-now-open-first-shipments-march-28/

[12] (2016) Oculus. Dostopno na: https://www.oculus.com/en-us/

[13] (2016) Oculus Rift – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Oculus_Rift

[14] (2016) Kinect for Windows Sensor. Dostopno na: https://msdn.microsoft.com/en-us/library/hh855355.aspx

[15] (2016) Kinect – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Kinect

[16] (2016) Leap Motion – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Leap_Motion

[17] (2016) Unity (game engine) – Wikipedia, the free encyclopedia. Dostopno na: https://en.wikipedia.org/wiki/Unity_(game_engine)

[18] (2016) Unity Asset Store. Dostopno na: https://www.assetstore.unity3d.com/en/

[19] (2016) The PubChem Project. Dostopno na: https://pubchem.ncbi.nlm.nih.gov/

[20] (2016) Leap Motion Developers. Dostopno na:
`https://developer.leapmotion.com/unity`

[21] (2016) Gloveone. Dostopno na: `https://www.gloveonevr.com/`